Manuel du développeur : Projet Patchwork

Introduction

Ce document décrit l'architecture de notre projet Patchwork, conçu et réalisé par Bryan PADJA et Achraf Quabil, des étudiants en première année de cycle d'ingénieur informatique à ESIPE. Le projet est organisé autour de sept classes principales qui permettent de créer, contrôler et visualiser une partie de Patchwork.

à l'attention de Mme. Béal et Mme. Van Den Bogaard.

Architecture de l'application

1. ChoiceLevel

La classe `ChoiceLevel` permet à l'utilisateur de choisir le niveau de la partie. Elle contient des méthodes pour créer et afficher une interface utilisateur, collecter les sélections de l'utilisateur et transmettre ces informations au reste de l'application.

2. Main

La classe `Main` est le point d'entrée de l'application. Elle crée les instances nécessaires des autres classes et démarre le jeu.

3. Patch

La classe `Patch` représente les différentes pièces du jeu. Chaque instance de cette classe est une pièce unique, avec ses propres caractéristiques comme la forme, la taille, et les points. Les objets `Patch` peuvent être placés sur le plateau de jeu par les joueurs.

Chaque pièce est définie par son coût, son temps, son revenu (en boutons), et une grille qui représente la forme de la pièce. Les pièces peuvent être placées sur le plateau de jeu, et chaque placement a un impact sur le score et la progression du joueur.

Dans cette classe, plusieurs champs privés stockent les informations sur une pièce :

- `cost` : Le coût de la pièce en boutons.
- `time` : Le coût de la pièce en temps.
- `income` : Le revenu que la pièce apporte au joueur en boutons.
- `grid` : Une grille 2D qui représente la forme de la pièce.
- `width` et `height` : La largeur et la hauteur de la grille, respectivement.

Le constructeur de la classe `Patch` initialise ces champs en fonction des paramètres fournis.

Plusieurs méthodes d'accès (getters) permettent d'obtenir les informations sur une pièce, par exemple `getCost()`, `getTime()`, `getIncome()`, `getGrid()`, `getWidth()`, et `getHeight()`.

La méthode `toString()` fournit une représentation textuelle de la pièce, y compris sa forme graphique, son coût et son revenu.

La méthode `squareOccupied(int x, int y)` vérifie si une cellule spécifique dans la grille de la pièce est occupée.

La méthode `patchOverlaps(Patch other)` vérifie si une pièce chevauche une autre pièce.

La méthode `isEmpty()` vérifie si une pièce est vide, c'est-à-dire qu'elle ne contient aucune cellule occupée.

La méthode `getSize()` renvoie la taille totale (la surface) de la pièce.

La méthode `getGridValue(int row, int col)` renvoie la valeur à une cellule spécifique dans la grille de la pièce.

La méthode `setGrid(boolean[][] newGrid)` met à jour la grille représentant la pièce.

Enfin, la méthode `rotate(int degree)` renvoie une nouvelle instance de `Patch` avec la grille tournée d'un certain degré. Cette méthode est utilisée pour permettre aux joueurs de tourner leurs pièces avant de les placer sur le plateau de jeu.

Ps : nous savons qu'on pouvaire ce fichier en Record.

4. Player

Ce fichier Java définit une classe `Player`, qui représente un joueur dans le jeu Patchwork. Il inclut plusieurs caractéristiques liées à la position d'un joueur dans le jeu, ses ressources et son statut. Analysons en détail les différentes parties du code.

Attributs de la classe `Player`

1. `Patchwork patchworkPlayer` : Il s'agit de la représentation du patchwork du joueur. Le patchwork est un aspect central du jeu et est probablement utilisé pour garder une trace des pièces de tissu que le joueur a acquises.
2. `String name` : C'est le nom du joueur.
3. `int buttons` : C'est le nombre de boutons que le joueur possède. Dans le contexte du jeu, cela peut être considéré comme le score du joueur.
4. `int position` : Il s'agit de la position actuelle du joueur sur le plateau de temps.
5. `boolean done` : Un drapeau pour indiquer si le joueur a terminé son tour.
6. `int timePlayer` : C'est le temps restant pour le joueur.

Constructeur de la classe `Player`

Le constructeur `Player(String name, Patchwork patchworkPlayer)` permet de créer un nouvel objet `Player` en initialisant le nom du joueur et son patchwork. Il initialise également le nombre de boutons à 5, la position à 0, le statut du tour à "non terminé" (`false`), et le temps du joueur à 0.

Méthodes de la classe `Player`

Il y a plusieurs méthodes dans cette classe pour obtenir et définir les attributs du joueur, y compris pour obtenir le nom du joueur, le patchwork du joueur, le nombre de boutons du joueur, la position du joueur, le temps restant du joueur et pour déterminer si le joueur a terminé son tour.

La méthode `setPosition(int position)` permet de définir la position du joueur sur le plateau de temps. Si la position donnée est négative, elle lance une `IllegalArgumentException`.

La méthode `addTimePlayer(int n)` est utilisée pour ajouter du temps au joueur.

La méthode `payButtons(int cost)` permet au joueur de payer un certain coût en boutons. Si le coût est négatif ou supérieur au nombre de boutons du joueur, elle lance une `IllegalArgumentException`.

La méthode `advanceAndReceiveButtons(Player other, TimeBoard timeBoard)` est utilisée pour faire avancer le joueur jusqu'à la position du jeton de l'autre joueur +1 et pour recevoir un bouton par espace déplacé.

La méthode `openPatch(Patch patch)` vérifie si un certain morceau de tissu est disponible pour le joueur.

La méthode `toString()` surcharge la méthode `toString()` de la classe `Object` et retourne une chaîne de caractères représentant le joueur, y compris son nom, son nombre de boutons et sa position sur le plateau de temps.

5. Patchwork

Ici, notre code implémente un jeu de patchwork où chaque joueur essaie de compléter un tableau de 9x9 avec des pièces de formes différentes. Le code utilise toujours une architecture orientée objet et est écrit en Java. Ci-dessous quelques détails :

1. Classe `Patchwork` : Cette classe représente le plateau de patchwork d'un joueur. Elle se compose d'un tableau de 9x9 de `Patch` et garde la trace des lignes et des colonnes complétées.
2. Variables de l'Instance : La classe `Patchwork` contient des variables privées pour représenter le tableau (`grid`), ainsi que le nombre de lignes (`fullRows`) et de colonnes (`fullCols`) complétées.
3. Constructeur `Patchwork()` : Ce constructeur initialise un nouvel objet `Patchwork` avec un tableau vide de 9x9 et aucun ligne ou colonne complétée.
4. Méthode `openPatch(Patch patch)` : Cette méthode vérifie si un patch donné est disponible pour être placé sur le plateau de patchwork. Si le patch est null, elle génère une `IllegalArgumentException`.
5. Méthodes `getFullRows()` et `getFullCols()` : Ces méthodes retournent respectivement le nombre de lignes et de colonnes complètes sur le plateau de patchwork.
6. Méthode `isFull()` : Cette méthode vérifie si le plateau de patchwork est complet (tous les 81 patches ont été placés).
7. Méthodes `isRowFull(int row)` et `isColFull(int col)` : Ces méthodes privées vérifient respectivement si une ligne ou une colonne donnée est complète (tous les 9 patches de la ligne ou de la colonne ont été placés).
8. Méthode `addToPatchwork(Player player, Patch piece, int row, int col)` : Cette méthode ajoute une pièce donnée au patchwork du joueur. Elle vérifie d'abord si le joueur et la pièce sont

valides, puis vérifie si les coordonnées sont valides. Si la pièce est ajoutée avec succès, le coût de la pièce est déduit des boutons du joueur.

9. Méthode `movePiece(Patch piece, int currentX, int currentY, int newX, int newY)` : Cette méthode déplace une pièce existante à de nouvelles coordonnées sur le plateau.
10. Méthode `getPlacedPieces()` : Cette méthode retourne une liste de tous les patchs actuellement placés sur le plateau.
11. Méthodes `getPieceX(Patch piece)` et `getPieceY(Patch piece)` : Ces méthodes retournent respectivement la position X et Y d'une pièce donnée sur le plateau.
12. Méthode `toString()` : Cette méthode retourne une représentation sous forme de chaîne du plateau de patchwork, y compris les patchs et leurs positions.

En résumé, cette classe représente le plateau de patchwork d'un joueur dans le jeu, et contient des méthodes pour manipuler et interroger l'état du plateau.

6. PatchworkGame

La classe `PatchworkGame` est clairement le coeur de notre jeu Patchwork. Elle gère le déroulement du jeu, les tours des joueurs, et la fin du jeu. Ci-dessous, voici en détail notre classe:

Variables Membres :

- `patches` : Une liste d'objets Patch qui représente les pièces de patchwork disponibles pour le jeu.
- `player1` et `player2` : Des instances de la classe `Player` représentant les deux joueurs du jeu.
- `size` : Une variable entière qui représente la taille du plateau de jeu.
- `timeBoard` : Une instance de la classe `TimeBoard`, un tableau pour stocker le plateau de temps.
- `fullGame` : Une variable booléenne qui détermine si le jeu est en phase 1 ou 2.

Constructeur :

Le constructeur `PatchworkGame(Player player1, Player player2, int size, boolean fullGame)` initialise les variables d'instance avec les valeurs passées en paramètres. Il charge également les pièces à partir d'un fichier si le jeu est en phase complète (`fullGame` est true), sinon il initialise les pièces pour un jeu simplifié.

Méthodes :

- `initializeOpenPatch()` : Cette méthode initialise la liste des pièces disponibles en fonction de la phase du jeu. Si `fullGame` est vrai, elle charge les pièces à partir d'un fichier. Sinon, elle initialise les pièces pour un jeu simplifié.
- `start()` : Cette méthode démarre le jeu. Elle fait jouer les tours pour les deux joueurs jusqu'à ce que le jeu soit terminé. Elle détermine ensuite le gagnant et affiche le résultat.
- `playTurn(Player player)` : Cette méthode fait jouer un tour pour le joueur donné. Elle invite le joueur à choisir une pièce de tissu à ajouter à son patchwork et à choisir l'emplacement sur le patchwork où le tissu sera placé. Elle met ensuite à jour les boutons du joueur, le temps, la position sur le tableau de temps, et définit le drapeau "done" à vrai pour indiquer que le tour du joueur est terminé.

- `loadDataPatch(Path path)` : Cette méthode charge les données des pièces à partir d'un fichier. Le fichier est supposé être au format CSV, avec chaque ligne représentant une pièce. Les informations sur chaque pièce comprennent le coût, le temps, le revenu, la largeur, la hauteur et la disposition de la grille.
- `isGameFinished()` : Cette méthode retourne vrai si le jeu est terminé, c'est-à-dire si les deux joueurs ont terminé leurs tours et s'il n'y a plus de pièces disponibles.

7. TimeBoard

La classe `TimeBoard` représente le plateau des joueurs. Elle gère les avancements des joueurs sur le plateau et permet de visualiser la progression du jeu.

Conclusion

Chaque classe dans ce projet joue un rôle spécifique dans la réalisation du jeu Patchwork. Maintenant, on comprend d'avantage l'utilité de l'architecture POO, cette dernière permet une organisation claire du code, facilitant le développement et la maintenance du projet.

Annexe

Améliorations et Corrections depuis la Soutenance Bêta

Depuis la soutenance bêta, notre chargé de TP Mme. Van Den Bogaard, nous a fait des retours concernant le jeu et son code. On a prit en compte tous les commentaires et suggestions reçus et avons apporté plusieurs améliorations et corrections importantes à notre projet. Voici quelques détails à ce sujet :

Meilleur TimeBoard

En se basant sur les retours, nous avons travaillé sur l'amélioration du TimeBoard, afin de corriger au mieux les bugs d'affichage.

Rotation des Pièces

A ce moement, on en avait pas encore implémenter. On a suivit les conseils de notre enseignante sur comment intégrer cette partie du jeu.

Gestion des Pièces Choisies

Il y avait un problème dans notre version bêta où les pièces choisies par les joueurs ne disparaissaient pas de la liste des pièces disponibles. Nous avons corrigé ce problème en s'assurant que chaque fois qu'un joueur sélectionne une pièce, elle est retirée de la liste des pièces disponibles.

Commentaires Accrus dans le Code

Aussi, nous avons fait un effort pour augmenter la quantité et la qualité des commentaires dans notre code.

Ajout d'Exceptions et de Préconditions

Suite à vos suggestions, nous avons ajouté plus d'exceptions et de préconditions à notre code.

Merci

Aux lecteurs du temps accordé à la lecture de ce manuel. Et merci à nos enseignants.

Achraf et Bryan.